REMARKS

Claims 1, 5, 6, 8-13 and 22-35 are currently pending in the subject application and are presently under consideration. A clean version of the claims is found at pages 2-5. Favorable reconsideration of the subject patent application is respectfully requested in view of the comments herein.

I. Rejection of claims 32-34 Under 35 U.S.C §102 (e)

Claims 32-34 stand rejected under 35 U.S.C. §102(e) as being anticipated by Kawachi, *et al.* (U.S. 6,690,981). Withdrawal of this rejection is respectfully requested for at least the following reasons.

Kawachi, *et al.* fails to disclose, teach, or suggest all the elements set forth in independent claim 32 (and claims 33-34 which respectfully depend there from).

> For a prior art reference to anticipate, 35 U.S.C. §102 requires that ***"each and every element as set forth in the claim is found***, either expressly or inherently described, **in *a single prior art reference.***" In re Robertson, 169 F.3d 743, 745, 49 USPQ2d 1949, 1950 (Fed. Cir. 1999) (quoting Verdegaal Bros., Inc. v. Union Oil Co., 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987)) (emphasis added). ***"The identical invention must be shown in as complete detail as is contained in the ... claim.***" Richardson v. Suzuki Motor Co., 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989) (emphasis added).

The present invention relates to a system and method for ***interacting with computer programming languages at a semantic and/or syntactic level*** through an object-oriented code model and interface. (*See* Abstract). Particularly, the claimed invention enables a computer programmer to interact with a plurality of programming languages through a code model interface wherein syntactically and functionally unique programming languages are modeled by objects that encapsulate semantics and functionality of a language. (*See* pg. 2, ln. 6-10).

In accordance with the present invention, a code model returns objects to a programmer that encapsulate high level or semantic programming elements of a computer programming language while insulating the programmer from the syntax of the high-level, semantic programming elements. (*See* pg. 2, ln. 11-14). Thus, programmers can view the semantic

elements of a language and utilize the semantic elements in a language without learning the syntax of that language. (*See* pg. 2, ln. 29-31).

In particular, as recited in independent claim 32, the subject invention provides for a method for interacting with a computer programming language comprising i) retrieving a code model object from a code model; ii) ***interacting with the code model object to specify a semantic element***; and iii) incorporating the code model object into a program project, ***the code model mapping the code model object to the correct syntax for a particular computer programming language associated with the semantic element.***

Kawachi, *et al.* fails to disclose, teach or suggest such claimed aspects of the subject invention. Instead, Kawachi *et al.* discloses a system and method for enabling user interface code to be encapsulated in a sub-program of a graphical program. (*See* Abstract). Specifically, Kawachi, *et al.* fails to disclose any interaction with a code model object to specify a semantic element as recited by item (b) of independent claim 32 as set forth in the subject Office Action. Also with respect to independent claim 32, Kawachi, *et al.* fails to teach or suggest ***mapping the code model object to the correct syntax for a particular computer programming language associated with the semantic element*** as recited in such claim.

Further regarding claim 32, Kawachi, *et al.* merely discloses creation of a graphical program or flow diagram utilizing various function nodes or icons to accomplish a desired result. Kawachi *et al.* merely discloses that an assembled graphical program may be compiled to produce machine language to accomplish a desired method or process. In other words, Kawachi *et al. **is silent regarding any interaction to specify a semantic element as well as any technique to map a code model object to the correct syntax for a particular computer programming language associated with the semantic element*** as recited in independent claim 32.

In the field of computer science, it is well known that the term "semantics" is frequently used to differentiate the meaning of an instruction from its format. Furthermore, the format, which generally refers to the spelling of language components and the rules controlling how components are combined, is most commonly known as the language's syntax. For example, if a command is misspelled, this is a syntax error. If, on the other hand, a legal command is improperly entered in the current context, it is a semantic error.

Here, Kawachi, *et al.* fails to disclose, teach, or suggest any reference to an interaction to specify semantic language elements as well as mapping to obtain the correct syntax in relation to a particular computer programming language associated with the semantic element.

In view of at least the above, it is readily apparent that Kawachi *et al.* does not anticipate or suggest the subject invention as recited in independent claim 32 (and claims 33-35 which respectively depend there from). Accordingly, this rejection should be withdrawn.

II. Rejection of Claims 1, 5, 6, 8-13 and 22-31 under 35 U.S.C. §103(a)

Claims 1, 5, 6, 8-13, and 22-31 stand rejected under 35 U.S.C. §103 (a) as being unpatentable over McInerney, *et al.* (U.S. 5,325,533) in view of Conner *et al.* (U.S. 5,428,792). It is respectfully submitted that this rejection should be withdrawn for at least the following reasons.

McInerney, *et al.* fails to disclose, teach, or suggest a ***language neutral interface*** that ***insulates a programmer from unique syntaxes*** associated with a plurality of programming languages, as recited by claim 1, and ***programmatic interface to interact*** with a plurality of computer programming languages at a semantic and syntactic level ***in a language neutral manner***, wherein the code model ***provides isolation between the programmatic interface and computer programming languages***, as recited by claim 22. Rather, McInerney, *et al.* merely discloses an incremental building process (*e.g.*, compiling and linking programs) that models a program as a collection of components (See Abstract). It is readily apparent that McInerney, *et al.* fails to disclose, teach, or suggest all the claim limitations.

The Office Action dated April 8, 2004 concedes that McInerney, *et al.* fails to teach that the code model encapsulates a plurality of programming languages and provides a language neutral interface thereto that insulates a programmer from unique syntaxes associated with the plurality of programming languages.

The Office Action dated April 8, 2004 contends that Conner *et al.* teaches encapsulating a plurality of programming languages and a language neutral interface. Applicants respectfully disagree with this assertion. Rather, Conner *et al.* is directed to a system for developing class libraries in connection with object-oriented programs. Conner *et al.* refers to this system as the System Object Model (SOM). (*See* col. 3, ln. 27-32). It is respectively submitted that Conner *et al.* is not directed to an object-oriented programming method which encapsulates a plurality of programming languages and provides a language neutral interface. Instead, Conner *et al.* is

directed to SOM which is described as a ***language-neutral object model used to create class libraries usable from any programming language.*** This model may be used in conjunction with a procedural or object-oriented programming method (See col. 6, ln. 22-26).

As disclosed in Conner *et al.*, ***a file containing a language neutral Object Interface Definition Language (OIDL) definition*** is input into a resident compiler. Next, the compiler converts the OIDL definition file into an intermediate form. The intermediate form is input into an emitter to convert the intermediate form of the file into binding that conforms with a target language. Next, the bindings are input to the target language compiler to generate object modules which may be link edited to create a loadable module. (See col. 2, ln. 8-18).

In light of the forgoing, it is respectfully submitted that Conner *et al.* fails to teach or suggest encapsulating a plurality of programming languages and a language neutral interface as claimed in the subject invention. Conner, *et al.* merely discloses creation of an intermediate form of a file which may later be compiled by a specific target language compiler to generate modules which may be link edited to create a loadable module.

With respect to independent claim 22, it is respectfully submitted, for at least the reasons set forth hereinabove, Conner *et al.* does not teach or suggest an interface which allows for an interaction with a programming language in a language neutral manner as claimed in the present invention.

For at least these reasons, it is readily apparent that McInerney *et al.* considered individually or together with Conner, *et al.* fails to disclose, teach, or suggest all the claim limitations. Accordingly, claims 1 and 22 (as well as claims 5, 6, 8-13 and 23-31 depending directly or indirectly thereon) are allowable and withdrawal of this rejection is respectfully requested.

Likewise, for at least the reasons set forth hereinabove, it is respectfully submitted that Kawachi *et al.* considered individually or together with McInerney, *et al.* fails to disclose, teach, or suggest all the claim limitations of claim 35. McInerney *et al.* does not make up for the aforementioned deficiencies of Kawachi *et al.* with respect to independent claim 32 (which claims 33-35 directly or indirectly depend there from) as discussed in the 35 U.S.C. § 102(e) analysis above. Therefore, the subject invention as recited in claim 35 is not obvious over the combination of McInerney *et al.* and Kawachi *et al.* Accordingly, it is believed that claim 35 is allowable and withdrawal of this rejection is respectfully requested.

In summary, to reject claims in an application under 35 U.S.C. §103, an examiner must show an unrebutted *prima facie* case of obviousness. A *prima facie* case of obviousness is established by a showing of three basic criteria. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. *See* MPEP §706.02(j).

The Federal Circuit has consistently held that

> "...***virtually all [inventions] are combinations of old elements.***" Therefore an examiner may often find every element of a claimed invention in the prior art. ***If identification of each claimed element in the prior art were sufficient to negate patentability, very few patents would ever issue.*** Furthermore, rejecting patents solely by finding prior art corollaries for the claimed elements would permit an examiner to use the claimed invention itself as a blueprint for piecing together elements in the prior art to defeat the patentability of the claimed invention. ***Such an approach would be 'an illogical and inappropriate process by which to determine patentability.'*** *In re Rouffet*, 149 F.3d 1350, 1357, 47 U.S.P.Q.2d 1453 (Fed. Cir. 1998) (*citations omitted*).

Moreover, in addition to the analysis set forth above, obviousness cannot be established by combining the teachings of the cited art to produce the claimed invention, absent some teaching or suggestion supporting the combination. Teachings of references can be combined *only* if there is some suggestion or incentive to do so. Here, neither the nature of the problem to be solved, the teachings in the cited art, nor the knowledge of persons of ordinary skill provide sufficient suggestion or motivation to combine the references.

Instead, the Final Office Action relies on improper hindsight in reaching an obviousness determination. The Federal Court has held that to imbue one of ordinary skill in the art with knowledge of the invention in suit, when no prior art reference or references of record convey or suggest that knowledge, is to fall victim to the insidious effect of a hindsight syndrome wherein that which only the inventor taught is used against its teacher. One cannot use hindsight reconstruction to pick and choose among isolated disclosures in the prior art to depreciate the claimed invention. *In re Fine*, 837 F.2d 1071, 5 U.S.P.Q.2d (BNA) 1596 (Fed. Cir. 1988) (citations omitted).

Here, even if the claimed elements were interpreted to exist in the cited prior art, McInerney, *et al.*, Conner, *et al.* and/or Kawachi *et al.* cannot be combined to make the present invention obvious because there is not proper suggestion or motivation to combine the references' teachings to create the subject matter of claimed invention.

CONCLUSION

The present application is believed to be in condition for allowance in view of the above comments and amendments. A prompt action to such end is earnestly solicited.

In the event any fees are due in connection with this document, the Commissioner is authorized to charge those fees to Deposit Account No. 50-1063.

Should the Examiner believe a telephone interview would be helpful to expedite favorable prosecution, the Examiner is invited to contact applicants' undersigned representative at the telephone number listed below.

Respectfully submitted,

AMIN & TUROCY, LLP

Himanshu S. Amin
Reg. No. 40,894

AMIN & TUROCY, LLP
24TH Floor, National City Center
1900 E. 9TH Street
Cleveland, Ohio 44114
Telephone (216) 696-8730
Facsimile (216) 696-8731